

Universidad de Costa Rica

Escuela de Matemática

Herramientas de Ciencias de Datos I (CA-0204)

Bitacora 2

Bot Hatchet

Profesor: Luis Juárez Potoy

Estudiantes:

Andres Zuñiga Mora (C38733)

Anthony Flores Rojas (C32975)

Amy Chen Wu (C32203)

Leonardo Vega Aragon (C38313)

Fecha: 2025-10-13

Tabla de Contenidos

1- Identifique los siguientes puntos para los datos con los que realizará su proyecto:	1
a. Características de los datos	1
b. Población de estudio	1
c. Muestra observada	1
d. Unidad estadística o individuos	2
e. Variables de estudio	2
2. Agregar las primeras 5 filas de su tabla de datos	3
3- Resumen de 5 números de las variables cuantitativas y analizar el mismo. . . .	5
1. Elo	5
2. Número de movimientos.	6
3. Tiempo movimiento medio.	6
4. Score.	6
5. Resultado.	6
4- Hacer al menos un gráfico que describa la distribución para cada una de las variables cuantitativas.	6
5- Hacer al menos dos gráficos que describan la relación entre las variables.	8
6- Hacer al menos un gráfico que muestre la distribución de las variables categóricas.	10
7- Identificar valores faltantes y posibles outliers.	12
8- Investigar técnicas que permitan subsanar los valores perdidos y outliers.	13
Referencias:	14
Temas planteados inicialmente	18
Tema escogido	19

1- Identifique los siguientes puntos para los datos con los que realizará su proyecto:

a. Características de los datos

Los datos provienen de partidas de ajedrez simuladas entre distintos modelos de bots, principalmente las hechas por el mismo bot que estamos diseñando “Hatchet”. Cada partida guarda sus datos con dos niveles de detalle:

- `data_juego.csv` : información resumida de cada partida (resultado general, estado final, cantidad de jugadas, condiciones de finalización)
- `data_fuerte.csv` : información detallada jugada por jugada (movimientos, tiempo medio de cada jugada, color, posiciones FEN, tablero final)

Los datos son mixtos (variables categóricas y numéricas), generados artificial, aleatoriamente, bajo condiciones controladas (parametros que manipulamos en el codigo). Estos datos se guardan en un archivo llamado `data_juego.csv` y `data_fuerte.csv` consecutivamente.

b. Población de estudio

La población está compuesta por todas las posibles partidas de ajedrez jugadas por Hatchet bajo configuraciones. Cada fila de los `dataFrame` incluye configuraciones iniciales, movimientos secuenciales (FEN) y resultados finales.

Ejemplos de variables dentro de la población:

- Tablas en tipo FEN (Forsyth–Edwards Notation): “4t1r1/p1p2pp1/.../4T1R1”
- Secuencias de movimientos : “Nf3 nf6 e4 e5 Bb5 a6”

c. Muestra observada

La muestra observada corresponde al conjunto de partidas efectivamente simuladas y almacenadas en los `dataframes`. Es un subconjunto representativo de la población, limitado por parámetros de simulación (modo, modelo, profundidad de búsqueda, número de movimientos, etc.). Actualmente no se considera el tiempo de ejecución como variable una variable específica,

sino general, tomando una media del tiempo de respuesta de cada partida, aunque se planea implementarlo en fases posteriores de optimización de la red neuronal.

d. Unidad estadística o individuos

Tanto para data_juego y data_fuerte la unidad estadística o individuos a estudiar son las partidas de ajedrez representadas en tipo FEN y con datos completos de ellas en casa csv.

e. Variables de estudio

Variables en data_juego:

- Modo : Categórica nominal : Tipo de simulación (Hatchet vs Hatchet, Hatchet vs motor externo)
- Modelo : Categórica nominal : Bot/Hatchet principal que juega
- Modelo_contrario : Categórica nominal : Bot/Hatchet oponente
- Finalizado : Booleana : Indica si la partida concluyó formalmente
- Jaque : Booleana : Estado final en jaque
- Mate : Booleana : Verdadero si terminó por jaque mate
- Ahogado : Booleana : Verdadero si terminó en tablas por ahogado
- Material : Booleana : Verdadero si hay material insuficiente para continuar
- Regla75 : Booleana : Verdadero si se aplicó la regla de 75 movimientos
- Repeticion5 : Booleana : Verdadero si hubo repetición de posición cinco veces
- Numero_de_movimientos: Cuantitativa discreta: Total de jugadas realizadas

Variables en data_fuerte:

- Modo : Categórica nominal : Tipo de simulación
- Modelo : Categórica nominal : Bot/Hatchet que juega
- Modelo_contrario : Categórica nominal : Bot/Hatchet oponente
- Color : Categórica nominal : Color con el que juega el modelo
- Resultado : Categórica nominal : Resultado de la partida
- Finalizado : Booleana : Si la partida llegó a estado terminal
- Elo : Cuantitativa discreta : Valoración Elo estimada según color y secuencia

- Jaque : Booleana : Estado registrado en jaque
- Mate : Booleana : Estado registrado en jaque mate
- Numero_de_movimientos : Cuantitativa discreta : Total de jugadas de la partida
- Movimientos : Texto estructurado : Secuencia de jugadas en notación algebraica
- Tiempo_movimiento_medio: Cuantitativa continua : Duración promedio por movimiento
- Tabla_final : Texto estructurado : Posición final en FEN
- Tabla_totales : Texto estructurado : Historial completo de posiciones FEN de la partida

Siendo Elo y Resultados las únicas dos no tomadas en cuenta para esta entrega.

2. Agregar las primeras 5 filas de su tabla de datos

En esta sección se presentan las primeras 5 observaciones de los dos DataFrames utilizados, data_juego (datos generales) y data_fuerte (datos específicos del desempeño del modelo).

```
# Establecer directorio de trabajo
setwd("C:\\Users\\Anthonny\\Documents\\Progamacion\\R\\UCR\\CA-0204\\Proyecto\\datos")

# Cargar los DataFrames
data.juego <- read.csv("data_juego.csv")
data.fuerte <- read.csv("data_fuerte.csv")

# Mostrar las primeras 5 filas en formato tabla
cat("Primeras 5 filas del DataFrame data_juego")

## Primeras 5 filas del DataFrame data_juego

head(data.juego, 5)
```

```
##           Modo  Modelo Modelo_contrario Finalizado Jaque  Mate Ahogado
## 1 bot_vs_bot_interno Hatchet1      Hatchet1      TRUE FALSE FALSE  FALSE
## 2 bot_vs_bot_interno Hatchet1      Hatchet1      TRUE FALSE FALSE   TRUE
## 3 bot_vs_bot_interno Hatchet1      Hatchet1      TRUE FALSE FALSE  FALSE
```

```
## 4 bot_vs_bot_interno Hatchet1 Hatchet1 TRUE TRUE TRUE FALSE
## 5 bot_vs_bot_interno Hatchet1 Hatchet1 FALSE FALSE FALSE FALSE
## Material Regla75 Repeticion5 Numero_de_movimientos
## 1 FALSE FALSE TRUE 219
## 2 FALSE FALSE FALSE 194
## 3 FALSE FALSE TRUE 225
## 4 FALSE FALSE FALSE 163
## 5 FALSE FALSE FALSE 240
```

Cabe recalcar que las siguientes líneas no son legibles si no se abren en un archivo .csv.

```
cat("Primeras 5 filas del DataFrame data_fuerte")
```

```
## Primeras 5 filas del DataFrame data_fuerte
```

```
head(data.fuerte, 5)
```

```
##          Modo  Modelo Modelo_contrario Color Finalizado Jaque  Mate
## 1 bot_vs_bot_interno Hatchet1 Hatchet1 white      TRUE FALSE FALSE
## 2 bot_vs_bot_interno Hatchet1 Hatchet1 white      TRUE FALSE FALSE
## 3 bot_vs_bot_interno Hatchet1 Hatchet1 white      TRUE FALSE FALSE
## 4 bot_vs_bot_interno Hatchet1 Hatchet1 white      TRUE  TRUE  TRUE
## 5 bot_vs_bot_interno Hatchet1 Hatchet1 white     FALSE FALSE FALSE
```

```
## Numero_de_movimientos
```

```
## 1          219
```

```
## 2          194
```

```
## 3          225
```

```
## 4          163
```

```
## 5          240
```

```
##
```

```
## 1 b4XNc6Xf4XNd4XBb2Xe6Xb5XBa3X
```

```
## 2
```

```
## 3 Nc3XNf6Xa4Xd6Xe3Xb5Xg3Xe5XQh5X
```

```

0-0XN2c3XRg8XQd5Xf6XRa4XQxg3+XKe2XBxf1+XRxf1XQg4Xh4Xbxa4XQc4Xg6XKd1XNc5XQxc5XQg5XNg3XQxg
## 4
## 5 h4Xc5Xe3XNc6XNe2XNh6XNbc3XNf5Xb3XNxh4XNa4Xd6XNxc5Xa6XNxb7XQd7XNg3XNf5Xc3XRg8XQe2Xa5
##      Tiempo_movimiento_medio                      Tabla_final
## 1              0.2978873          k6r/8/8/8/8/6K1/7r/7r b - - 27 110
## 2              0.2738773              7k/8/8/8/8/5p2/8/r5bK w - - 0 98
## 3              0.2742453          7R/8/k7/n7/8/8/8/6K1 b - - 20 113
## 4              0.3078043      5Q1k/8/6P1/4p2p/p7/8/8/K5bB b - - 0 82
## 5              0.2612336 5k1B/8/5P1K/2P4p/4p2P/4P2B/8/8 w - - 9 121
##
## 1
## 2
## 3
## 4
## 5 rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1Xrnbqkbnr/pppppppp/8/8/7P/8

```

3- Resumen de 5 números de las variables cuantitativas y analizar el mismo.

Por la manera en la que se diseñó el presente proyecto, no se ha hecho uso de un dataframe externo que contenga variables cuantitativas. Por ende, las variables cuantitativas que se explicarán a continuación son aquellas que aparecerán en los dataframes próximos a generar:

1. Elo

Según la página Chess.com, la puntuación Elo es un número basado en la actuación de un jugador en partidas evaluadas disputadas anteriormente. Básicamente, mide la fuerza relativa de un jugador. Este sistema funciona de tal manera que premia en mayor magnitud a los jugadores más novatos (ganan más puntos) y castiga de mayor manera a los jugadores más expertos (pierden más puntos). Esta variable toma valores enteros no negativos. Es importante resaltar que el mayor elo perteneciente a una persona en 2025 es de 2839, mientras

que el bot con un mayor elo tiene un total de 3643. Estos dos valores, entonces, puede servir como parámetros para comparar el desempeño que muestren los bots entrenados durante el desarrollo de este proyecto.

2. Número de movimientos.

Esta variable toma el valor de un entero no negativo y representa el número de movimientos que realizó un jugador o un bot durante la realización de un juego.

3. Tiempo movimiento medio.

Esta variable representa la media de segundos que duró el jugador o el bot en hacer cada uno de sus movimientos durante la realización de un juego.

4. Score.

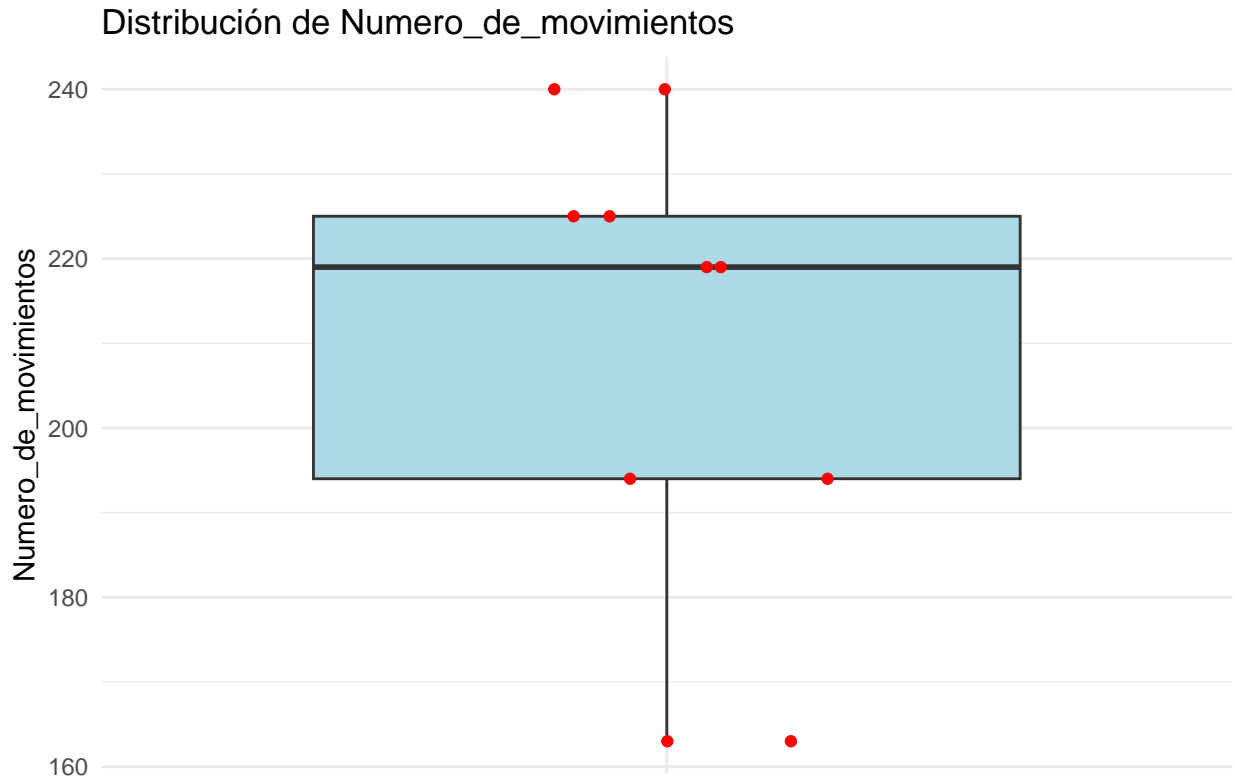
Esta variable toma valores de -1 a 1 y es utilizada con el fin de penalizar o premiar los movimientos que haga el bot durante su entrenamiento. Básicamente, antes de hacer cualquier movimiento el bot analiza todas las posibles jugadas y elije la que tenga una variable 'score' más alta. Cada una de esas variables va cambiando dependiendo del turno (pues después de cada turno se le añade una ligera cantidad de "ruido" a cada variable score) o de los movimientos del bot. Por ejemplo, si una jugada se repite tres veces seguidas entonces su score disminuye. Esta variable no se llega a mostrar al usuario en ningún momento.

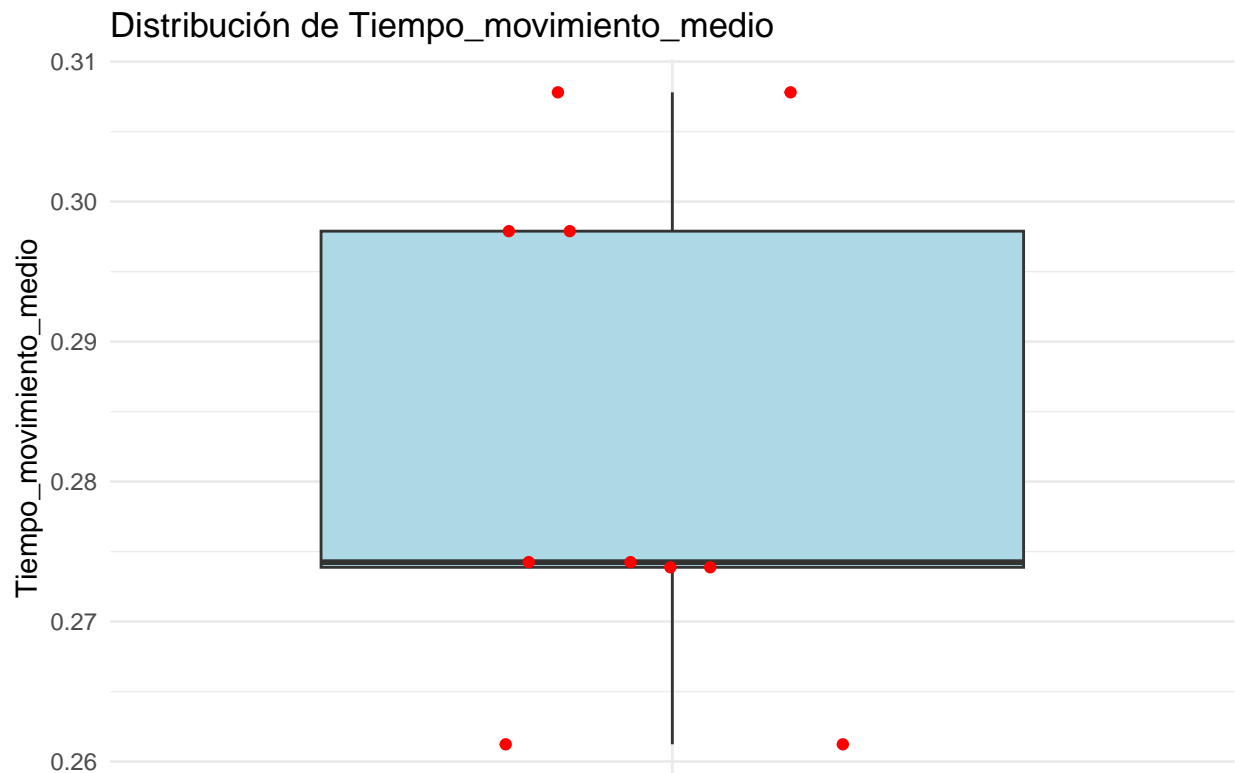
5. Resultado.

Esta variable toma valores de 0, 0.5 ó 1. 0 significa que el jugador en cuestión perdió la partida, 0.5 significa que quedó empate y 1 significa que ganó el juego.

4- Hacer al menos un gráfico que describa la distribución para cada una de las variables cuantitativas.

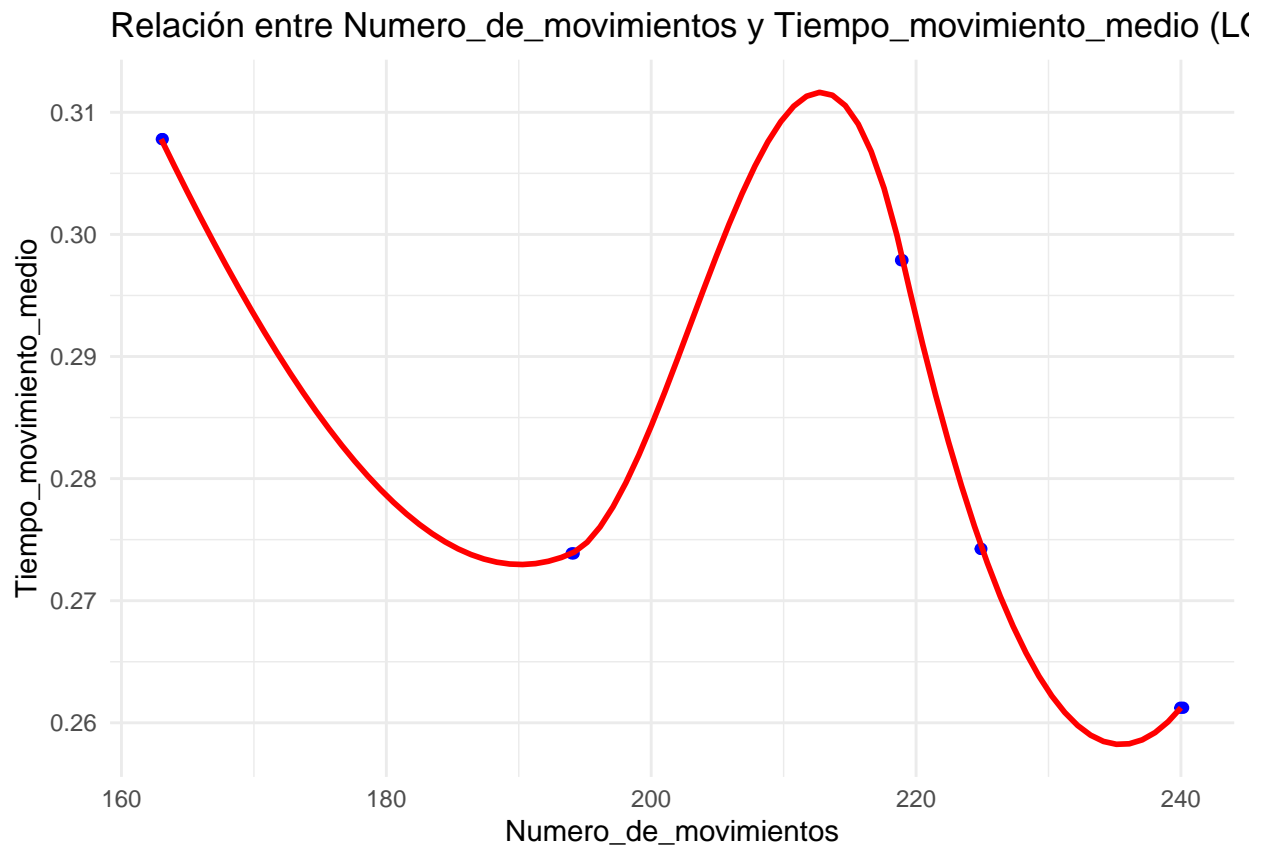
```
# Variables cuantitativas  
quant.vars.fuerte <- c("Numero_de_movimientos", "Tiempo_movimiento_medio")  
  
# Graficar distribucion  
plot.quantitative.distribution(data.fuerte, quant.vars.fuerte)
```



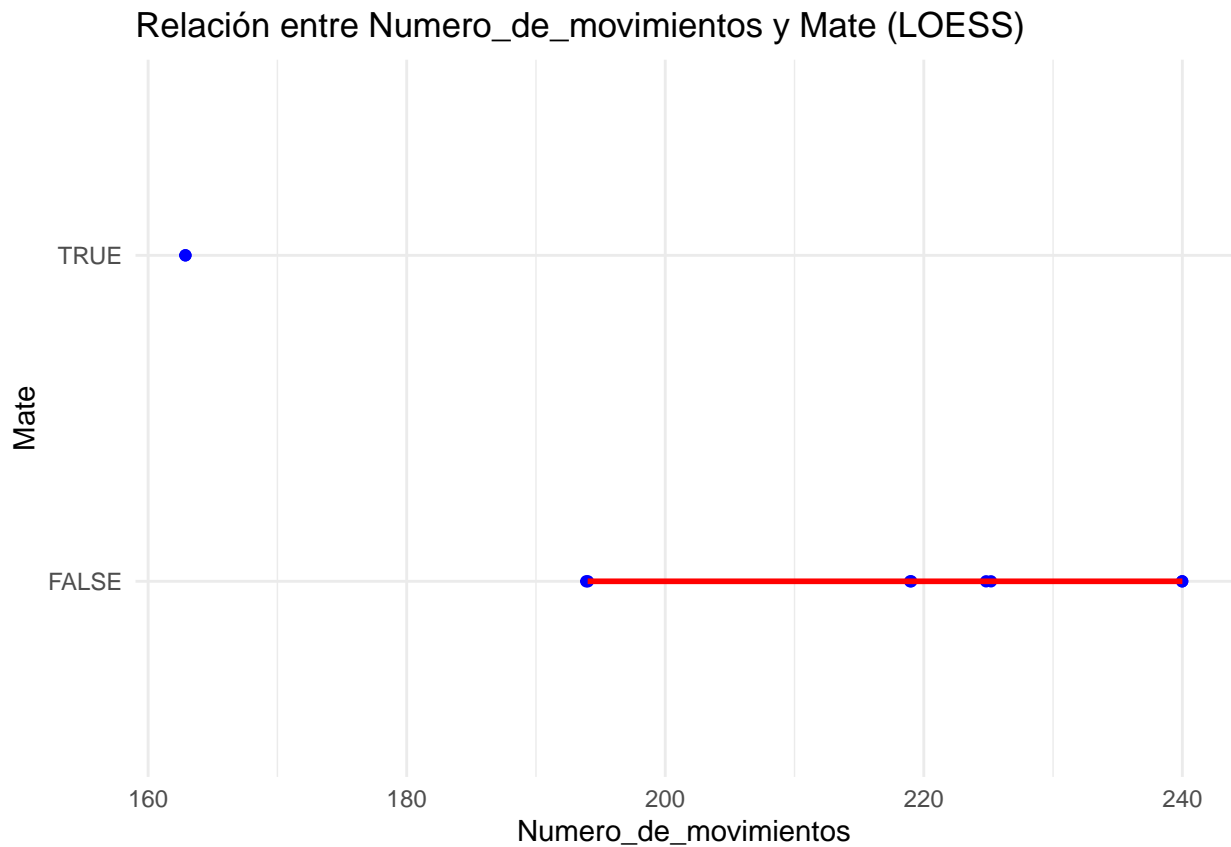


5- Hacer al menos dos gráficos que describan la relación entre las variables.

```
# Graficar relacion entre Numero_de_movimientos y Tiempo_movimiento_medio  
plot.quantitative.relationship(data.fuerte, "Numero_de_movimientos", "Tiempo_movimiento_medio")
```



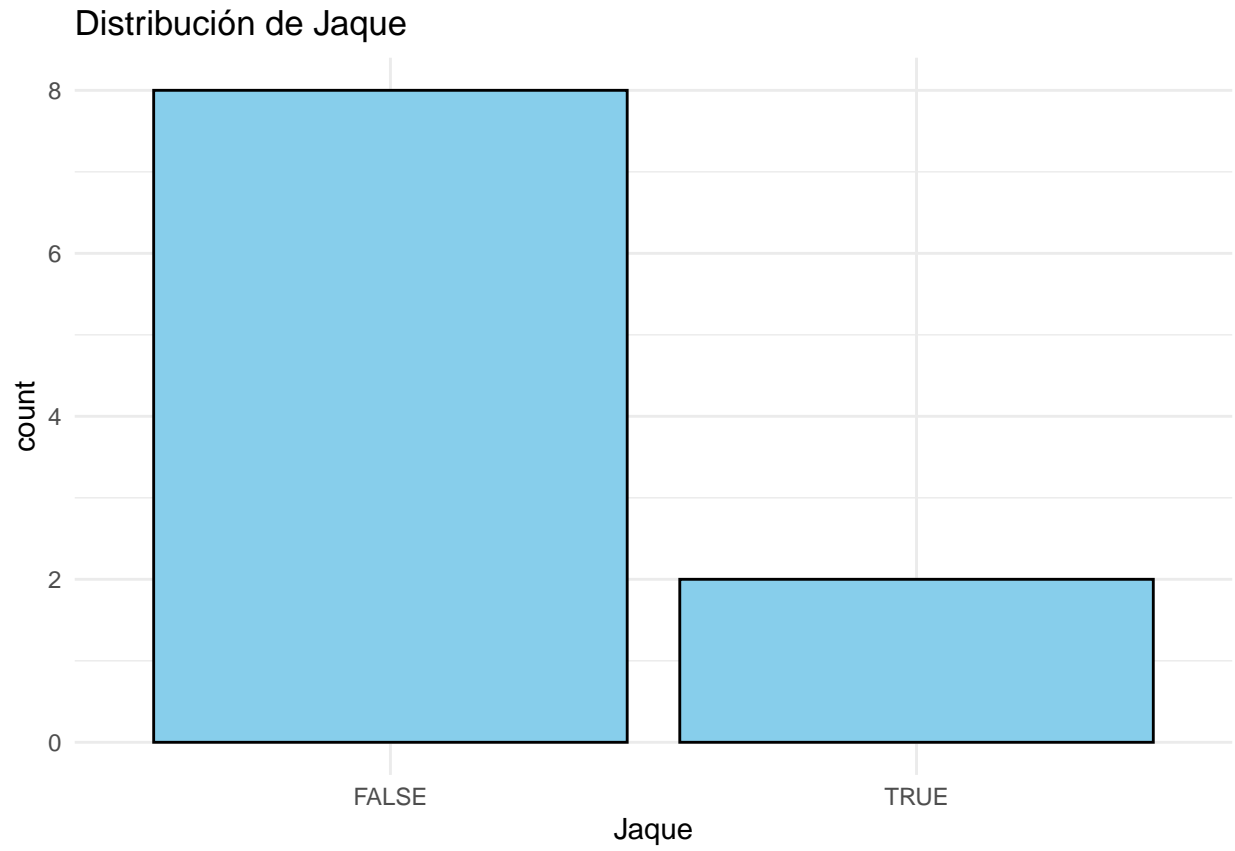
```
# Graficar relacion entre Numero_de_movimientos y Tiempo_movimiento_medio  
plot.quantitative.relationship(data.fuerte, "Numero_de_movimientos", "Mate")
```

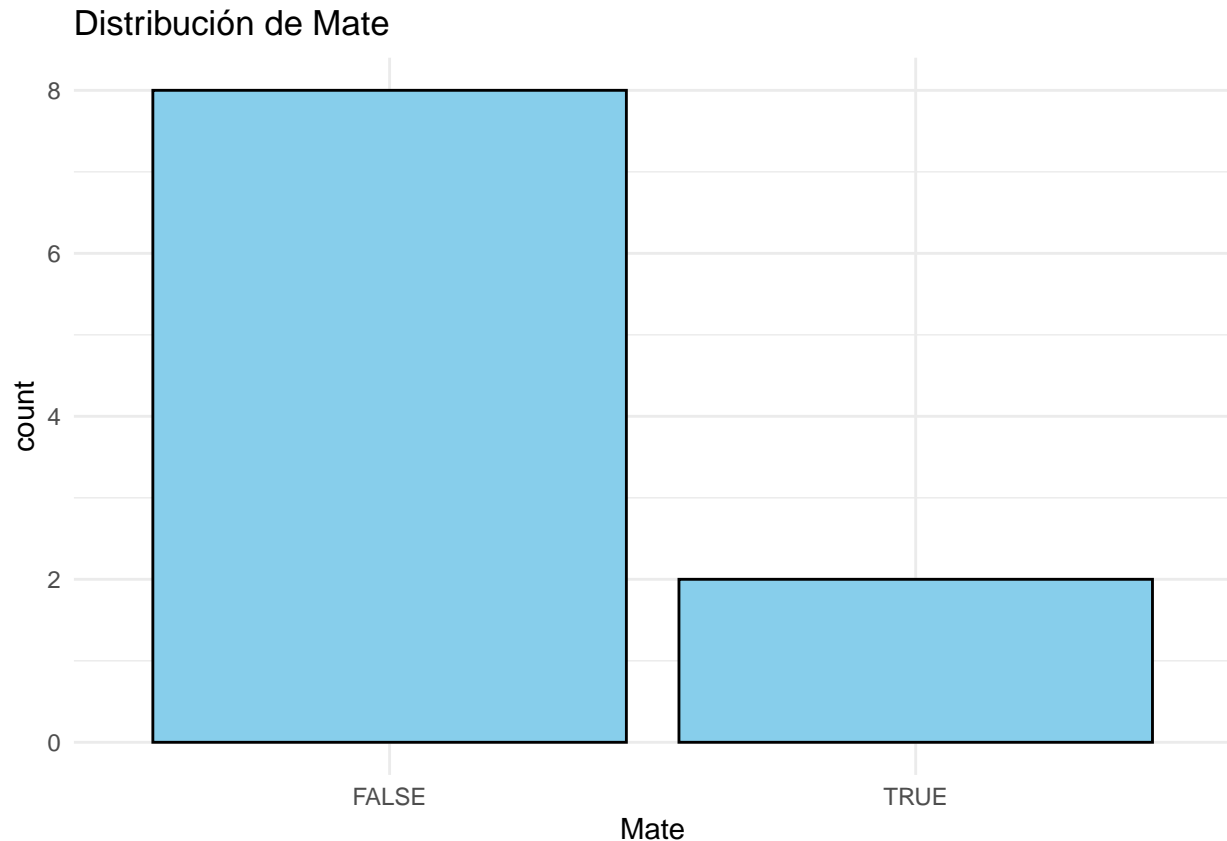


6- Hacer al menos un gráfico que muestre la distribución de las variables categóricas.

```
# Variables categóricas
cat.vars.juego <- c("Jaque", "Mate")

# Graficar distribución categórica
plot.categorical.distribution(data.juego, cat.vars.juego)
```





7- Identificar valores faltantes y posibles outliers.

Para este punto cabe recalcar que no contamos con valores faltantes, ya que los datos son autogenerados, y los valores faltantes serían generados por errores computacionales, y de igual forma, si los datos importantes de una partida se pierden, la partida entera se va a descartar. Si es un dato secundario (informativo) se perdiese solo se va a tomar la media (en el caso de número de movimientos, etc.) o el dato más abundante (en casos como el color con el que está jugando el bot, etc.), ya que la media nos va a permitir preservar la data lo más impoluta posible, y en el caso del dato más abundante, nos permite tomar una generalización de cada partida; y por el momento no se han considerado outliers, depende totalmente de las decisiones que se vayan realizando en el código para mejorar la escogencia y tiempo de respuesta de cada movimiento, ya que por el momento, para mostrar los datos en esta bitácora, se seleccionó un máximo de 240 movimientos. Ya el número de movimientos es un dato que limitamos en el código para evitar cálculos innecesarios de partidas a punto de

finalizar, o partidas sin fin. Reemplazar o descartar estos outliers no se ve como algo factible, ya que cada dato fuera de lo normal recalca una importante corrección en el código.

Cabe recalcar que estos outliers no son de gran importancia para el proyecto, ya que por ejemplo, las tablas por repetición de 5 movimientos de una misma pieza son muy comunes al inicio de los modelos. Este tipo de errores los hemos ido corrigiendo, implementando medidas como tener en cuenta el historial de las posiciones por partida. Y en este tipo de data, los outliers son totalmente dependientes de los metodos a aplicar a futuro para una optimizacion de la red neuronal y la escogencia del mejor movimiento.

8- Investigar técnicas que permitan subsanar los valores perdidos y outliers.

Como se especificó anteriormente, los tratamientos para valores perdidos y outliers son simples. Para los valores perdidos, si son de suma importancia, la partida se descarta; si son secundarios, se reemplazan por la media o el valor más abundante. Esta estrategia se fundamenta en la simplicidad y eficacia de métodos como la imputación por media o moda, que son comúnmente utilizados en análisis exploratorios debido a su facilidad de implementación y rapidez. Según Alam, Ayub, Arora y Khan (2023), técnicas como la imputación por media, mediana o moda pueden ser útiles en ciertos contextos, aunque pueden introducir sesgos si no se consideran las relaciones entre las variables y la naturaleza de los datos. En el contexto de este proyecto, donde los datos son autogenerados y no se han identificado valores faltantes significativos, la elección de imputar por la media o el valor más frecuente (moda) permite mantener la integridad del conjunto de datos y conservar la máxima cantidad de información sin afectar la validez del análisis.

En cuanto a los outliers, por el momento no se han considerado debido a que cada dato fuera de lo normal puede señalar una corrección importante en el código. Además, estos outliers no son de gran importancia para el proyecto, ya que, por ejemplo, las tablas por repetición de cinco movimientos de una misma pieza son muy comunes al inicio de los modelos. Este tipo de errores los hemos ido corrigiendo implementando medidas como tener en cuenta el historial de las posiciones por partida.

Referencias:

- Klein, D. (2022). Neural networks for chess: The magic of deep and reinforcement learning revealed. arXiv. <https://arxiv.org/abs/2209.01506>
- Aggarwal, C. C. (2018). Neural networks and deep learning: A textbook. Springer.
- Bishop, C. M. (2006). Pattern recognition and machine learning. Springer Science+Business Media.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419), 1140–1144. <https://doi.org/10.1126/science.aar6404>
- Alam, S. (2023). An investigation of the imputation techniques for missing data. ScienceDirect. Recuperado de <https://www.sciencedirect.com/science/article/pii/S2772662223001819>
- DeLeo, M., & Guven, E. (s.f.). Learning chess with language models and transformers. Whiting School of Engineering, Johns Hopkins University.
- R Core Team. (s.f.). CRAN: Manuals – The Comprehensive R Archive Network. Recuperado de <https://cran.r-project.org/manuals.html>

Anexo

Bitacora 1

UCR

Escuela de Matemática

Herramientas de Ciencias de Datos I (CA-0204)

Bitacora 1: Escogencia de tema para el proyecto grupal

Profesor: Luis Juárez Potoy

Estudiantes:

Andres Zuñiga Mora (C38733)

Anthony Flores Rojas (C32975)

Amy Chen Wu (C32203)

Leonardo Vega Aragon (C38313)

Fecha: 2025-10-13

Temas planteados inicialmente

- Simulación de interacción entre partículas subatómicas
- Criptografía con redes neuronales para seguridad de datos
- Modelado de probabilidades y estrategias en Póker mediante cadenas de Markov
- Modelado de epidemias con movimiento Browniano
- Desarrollo de un bot de ajedrez utilizando algoritmos de autoaprendizaje con jugadas aleatorias

De los temas propuestos, se valoró tanto la dificultad técnica como la aplicabilidad dentro del entorno de R, además de su conexión con problemas reales. La simulación de interacción entre partículas subatómicas generó interés por su trasfondo físico, pero fue descartada rápidamente: exige un manejo avanzado de mecánica cuántica (el cual ninguno de los compañeros posee), interacción de fuerzas fundamentales y modelos de física computacional que sobrepasan el alcance del proyecto. Su nivel de formalismo matemático y de teoría de campos demanda un tiempo que no es factible. En el caso de la criptografía con redes neuronales, se enfrentó el mismo obstáculo: la complejidad de integrar aprendizaje profundo con protocolos de seguridad de datos exige un dominio sólido de machine learning aplicado a criptografía moderna, lo cual rebasa el margen de trabajo. El tema de modelado de epidemias con movimiento Browniano también se dejó de lado, ya que aunque su atractivo matemático y probabilístico es innegable pero, no se encontró una forma clara de vincularlo con aplicaciones prácticas inmediatas que justificaran el esfuerzo de simularlo en R en el contexto del curso. Por otro lado, el modelado de probabilidades y estrategias en Póker mediante cadenas de Markov fue evaluado como viable, pero perdió peso frente a otras opciones por no aportar tanto en términos de conexión con algoritmos de autoaprendizaje contemporáneos (redes neuronales). Finalmente, el desarrollo de un bot de ajedrez con algoritmos de autoaprendizaje y jugadas aleatorias destacó como la opción más completa: accesible, con suficiente rigor matemático y alineada con campos como teoría de juegos, probabilidad y aplicaciones en inteligencia artificial, convirtiéndose en la elección final.

Tema escogido

El tema escogido fue Desarrollo de un bot de ajedrez utilizando algoritmos de autoaprendizaje con jugadas aleatorias. La motivación central de esta selección radica en que el ajedrez, por ser un juego de estrategia universalmente estudiado, resulta idóneo para probar y aplicar metodologías de autoaprendizaje como Q-learning o redes neuronales. La lógica del bot consiste en iniciar con jugadas aleatorias y, a partir de la retroalimentación de recompensas y penalizaciones, ajustar progresivamente sus decisiones hasta alcanzar patrones estratégicos más eficientes.

Este enfoque conecta de manera directa con escenarios del mundo real, dado que el ajedrez encapsula conceptos de estadística, probabilidad, teoría de juegos y toma de decisiones bajo incertidumbre. En ese sentido, la implementación en R no solo permite modelar dinámicas complejas del juego, sino también extrapolar los resultados a contextos de optimización y aprendizaje automatizado presentes en inteligencia artificial y ciencia de datos aplicada.

La elección final se fundamenta en su balance entre factibilidad técnica y profundidad matemática. Este tema no solo abre la posibilidad de evaluar probabilidades de éxito en distintas jugadas, sino que también permite diseñar y refinar estrategias que maximicen el rendimiento del bot a partir de una base inicial aleatoria. Con ello, se obtiene un proyecto que combina un ejercicio mas practico con un aporte conceptual en las matematicas y algoritmos de autoaprendisaje.